ITML Message and Protocol Specification

Working Draft

Last Updated: 11/22/00

Control Number:
Version: 0.8

Editor(s): David Orchard

Jamcracker, Inc.

ITML Vision

ITML - the Information Technology Markup Language - is a set of specifications of protocols, message formats and best practices in the ASP and ASP aggregation market to provide seamless integration of partners and business processes. It is based on open standards, particularly XML and HTTP. It also uses emerging standards, particularly SOAP and XML Schema.

Abstract

This document provides a framework for specific interactions to occur between Jamcracker and an ASP. An example interaction is a User Provisioning request. Each set of interactions is known as an ITML Best Practice. This document is a companion document to each Best Practice specification.

This specification describes the following key decisions:

- XML Schema is the type specification language
- message format is SOAP
- a SOAP error structure
- a set of protocol errors
- encoding rules for graphs of data
- encoding rules for methods
- namespaces standards in messages
- multi-part message encoding
- HTTP Binding including Authentication

Status of this Document

This document is a Working Draft, issued by the Jamcracker ITML team, for review by selected partners.

The ITML team expects that significant changes will occur in this document before version 1.0 is released. The ITML Team will not allow early implementation to constrain its ability to make changes to this specification prior to final release.

Specification Status

This specification is incomplete in some regards. The samples and schemas are fragments only, without the SOAP enveloping information. The use of the SOAP Schema must be integrated with the samples and schemas. This is delayed because of the lack of multiple namespaces in authoring tools, specifically XML Spy

Relationship to other standards

The ITML Framework 1.0 has been influenced by many recent standards efforts including, but not limited to, the following:

Normative

XML:

http://www.w3.org/TR/REC-xml

XML Namespaces
http://www.w3.org/TR/REC-xml-names/

SOAP:
http://www.w3.org/TR/SOAP/

XML Schema Structures:
http://www.w3.org/TR/xmlschema-1/

XML Schema Data Types:
http://www.w3.org/TR/xmlschema-2/

The MIME Multipart/Related Content-type
http://www.ietf.org/rfc/rfc2387.txt

URL
http://www.ietf.org/rfc/rfc1738.txt

Non-Normative

BizTalk:
http://msdn.microsoft.com/xml/articles/biztalk/biztalkfwv2draft.asp

MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)
http://www.ietf.org/rfc/rfc2557.txt

Content-ID and Message-ID Uniform Resource Locators:
http://www.ietf.org/rfc/rfc2111.txt

SOAP Messages with Attachments:
http://static.userland.com/weblogsCom/gems/soapweblogscom/soapMessagesWithAttachments.html

ebXML TRP Envelope specification
http://www.ebxml.org/specdrafts/Envv0-5.pdf

XML Schema Primer
http://www.w3.org/TR/2000/WD-xmlschema-0-20000407/primer.html

Audience

This document is a technical specification and is intended for developers and architects.

Document Conventions

The following notations are used to present material in this document:

ISSUE: An issue is a direct request for feedback from the audience. An issue reflects a lack of decision due to insufficient or conflicting inputs. These are resolved through the acquisition of more input

NOTE: Extra normative information that the author(s) wish to draw the attention of the reader to.

Use Cases and Requirements

The following Use cases describe the interactions supported by this specification.

Create ITML Best Practice

1. An author creates an ITML Best Practice document referencing this document. This requires specifying use cases, requirements, document schemas, specific interaction sequences, namespaces, and errors

Requirements

1. ITML must be straightforwardly usable over the Internet.
2. The ITML expression language must be XML.
3. The ITML design must be prepared quickly.
4. The ITML design must be formal, concise, and illustrative.
5. ITML instance documents must be human-readable and human-writable.
6. ITML must be feasible to implement.
7. ITML must utilize existing standards, such as HTTP and URIs.
8. ITML must allow for protocol specific security mechanisms
9. ITML must allow partner specific extensions for requests and responses.

Terminology

ASP – An application provided over the net and typically charged by the month. Note that there is no technical difference between an ASP and a web site. An ASP can provide a single or multiple business processes.

ASP Integration – A platform for Collaborative Business Processes, typically offering business processes that span ASPs.

Authorized User – A user that has appropriate credentials for accessing the protected resource.

Browser Web Services – Simply a URL representing a unit of work consumed by web browsers.

Business Process – A workflow consisting of a series of services, either web services or browser web services. A Business Process typically has an implicit state transition definition. A business process is typically the implementation of a use case.

Collaborative Business Process - A Collaborative Business Process (CBP) is a process supporting online collaboration among business partners. Definition by RosettaNet.

ITML(Information Technology Markup Language) – a set of XML specifications for interactions between Application Service Providers focused on the outsourced IT industry

ITML Best Practice – a specification describing a specific interaction(s) between Jamcracker and a partner. An example is user provisioning.

ITML Document – a set of XML compliant constructs that conform to ITML. An example is an Address element

ITML Message – an ITML Instance document encoded in the ITML Message and Protocol format and send over an ITML accepted protocol. An example is an ITML User Provisioning request encoded in the ITML Message and Protocol format (SOAP) and sent over HTTP.

Web Service – A single method or procedure accessed via XML and a protocol, such as SOAP. Definition by MSFT, IBM, et al as part of SOAP.

ITML Message Structure

ITML requests consist of actions upon objects.
ITML Messages are SOAP based messages, as defined by the SOAP 1.1 specification.

Request

The following shows a sample of a user profile request.

```
<SOAP-ENV:Envelope
                    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <SOAP-ENV:Body>
      <prov:GetUserProfile xmlns:prov="http://www.itml.org/ns/provisioning/request">
                    <userid>dchen</userid>
                    <company>jamcracker</company>
       </prov:GetUserProfile>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request Body Contents

The contents of the Body element are specified in a particular best practices specification.

Response

The following shows a sample response fragment.

```
<SOAP-ENV:Body>
      <prov:GetUserProfileResponse
xmlns:prov="http://schemas.itml.org/ns/provisioning/request">
       <User>
       <PersonalInfo>
          <firstName>David</firstName>
          <middleInitial>B</middleInitial>
          <lastName>Orchard</lastName>
          <Salutation>Mr.</Salutation>
          <SSN>123-12-1234</SSN>
          <DateOfBirth xsi:type="date">01/02/1234</DateOfBirth>
          <Gender>M</Gender>
        </PersonalInfo>
…..
      </User>
      </prov:GetUserProfileResponse>
   </SOAP-ENV:Body>
```

Errors

SOAP defines a FAULT element with a faultcode, faultstring, optional faultactor, and optional detail elements. A detail element is always required when an error in processing the body occurs. The error may be an ITML error or it may be a document specific error. The ITML error codes ranges are defined in the ITML-ERR or the best practices specific namespace. The detail element may contain a detailList element with multiple detail elements if mulltiple errors can be reported.

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ITML Provisioning addUser Response Sample -->
<prov:addUserResponse xmlns="http://www.itml.org/ns/provisioning"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ITMLProvMethods.xsd">
    <ITMLFaultDetail>
        <faultcode>
            prov:InvalidUser
        </faultcode>
        <faultstring>
            Invalid Document
        </faultstring>
    </ITMLFaultDetail>
</prov:addUserResponse>
```

The ITML-ERR namespace defines the following values:

- InvalidXMLDocument
- VersionMismatch
- ApplicationFailure
- RequestTimedOut
- Unauthorized
- UnknownCommand

Each ITML Best Practice will define errors in its namespace. An example might be a prov:InvalidUser error code in the provisioning namespace. This does not change the format of the fault element.

NOTE: The SOAP specification does not provide for easy representation of multiple errors. Indeed, the SOAP specification does not even allow for extension of the soap Fault element. It is planned to raise these requirements to the XML Protocols Working Group

ITML Encoding rules

The encoding rules for ITML messages specifies how each message content is created from the information model. The desire is to send the minimal set of the information in the request, balanced against having too many request types. The base units of the information model must always be sent completely, such as Actor, Company, etc.

The graph of the information model that is sent should be passed by value rather than pass by reference. An example of this is passing an actor with address information being passed as an actor with address elements inside, rather than an actor referring to address(es). The state of each object and the relationship between them is encoded as a state object. The state is considered.

Whenever there is a relationship that is being modified, each of the objects and the state change is encoded as a separate object, ie addUser to company, add service to company, add service to user, modify service for user, etc. For example, an update service for company request has 3 parameters: company, service, and state. Combined with the request, this can be though of as verb(subject, object).

Data propagation is accomplished using 3 methods: add, update, and delete.

There are a few exceptions to this rule. In particular, a user identity contains references to companies and therefore are bundled together rather than separately.

Whenever a relationship is being swapped, the old and the new reference must be present in the request, ie swap service A to service B for user C requires A, B, and C to be present. This is encoded using the update method.

Naming Standards

Namespaces

ITML documents use namespaces. The namespaces will constructed according to the emerging namespace standards:

1. a short namespace identifier
2. the identifier is also in the namespace value at the end
3. an ns subdirectory is used. **NOTE:** Microsoft uses a schema of schema.xyz.org instead.

A sample is xmlns:company="http://www.itml.org/ns/company "

There will be many elements per namespace within the ITML and Jamcracker hierarchies. The namespaces should be roughly the level of Java packages.

Requests are under the namespace of the logical subystem that the request belongs to, ie xmlns:prov="http://www.itml.org/ns/provisioning/"

Partner specific namespaces are encoded by appending company name in the URI, ie xmlns:mm="http://www.itml.org/ns/managemark/"

XML Syntax

XML elements are upper camel-case naming convention. That is elements begin with a capital letter, move to lower case for the rest of the first word. The next word in an element follows the same convention. Acronyms are usually all upper case.

Abbreviations are not to be used in elements or attributes unless they are commonly accepted practice or are acronyms. For example, UserID is acceptable but StateAbbr is not.

In general, elements based upon a complex type will have the containing element prefixing the name. For example, a NameInfo element inside a Company element will actually be called CompanyNameInfo. This is because a number of tools do not support local element name definitions, only global element name definitions.

All requests and responses schemas should be in a single file. All data structures should be in a single file, included by the request/response schema file. Early attempts at provisioning used an Address.xsd, Name.xsd, User.xsd, and Company.xsd. However, tool support for nested includes appears to be buggy. For example, User includes Address. An ASP specific User that includes User causes Address to be included twice, which the tooling views as a redeclaration of the User data types.

Versioning and Change Management

This specification does not provide for an explicit version specification above and beyond the underlying protocols.

Exceptions

No exceptions to the SOAP or XML Schema specifications are currently specified

Extensions

Some of the ITML Best Practices specifications, such as provisioning, do not completely meet the needs of all of Jamcracker Partners. The ITML Message And Protocol document allows for extensions to the existing Best Practices on a per-partner basis.

Best Practices specifications can allow for extensibility through the XML Schemas support extension (inheritance) of elements. XML Schema also supports equivalence classes, that the ability of one element to act as another.

Per-partner extensions to a particular ITML Best Practices document is done through a per-partner XML Schema file. The extensions may be done on requests or responses. The extension elements must occur after the required content. The extensions will be made without modification to the ITML Best Practices schema documents. For example, and addUser request has a User parameter. The extension mechanism means that the addUser has a User Parameter followed by partner specific elements. This causes the undesired side effect that the content may be somewhat unintuitive. Please refer to the ITML Provisioning Specification for a sample of this mechanism

ITML Messaging and Protocol allows for extension through non-ITML protocols as well. A partner may require the use of a particular encoding of information in addition to ITML messages. An example is an OTA encoded message. Where the protocol only defines a message format – and not a transport – then ITML supports multi-part messages. Thus an ITML provisioning request and an OTA provisioning request could be bundled together.

Multi-part messages

ITML Messages are encoded according to the MIME Multipart/Related Content-type
http://www.ietf.org/rfc/rfc2387.txt

This uses the Content-ID and Message-ID Uniform Resource Locators.

When documents must be sent that cannot be encoded as SOAP - typically vocabularies such as WML, MathML, XML Schema and non-xml content – these are then encoded as ebXML documents. That is, there will be a SOAP Header section in one part, and another part with the main document(s). There is no Base64 encoding or Cdata encoding of content.

Security

General security is not directly addressed by ITML. Authorization and audit trails are the purvey of the Jamcracker Platform and the partner web site. Authentication, non-repudiation, message integrity and message security are defined on a per-protocol basis.

Authorization of requestors to issue requests – the classification into authorized user – that result in Jamcracker/Partner interactions is left to Jamcracker or the Partner. For example, the authorization of users that can create ITML Provisioning Add User requests, which result in Add User requests to partners, is performed by Jamcracker and not part of the specification effort.

Protocol

ITML is a stateless request/response protocol. It is stateless as all information is passed in the request. There is no use of cookies, tokens or sessionIDs to force state on a partner site. The default transport protocol is HTTP. Other protocols – such as JMS and SMTP– may be used.

HTTP Exchange

The following describes a set of interactions occurring using HTTP between Jamcracker (JC) and an ASP (A).

1. JC creates an HTTPS connection (C) to a predetermined URL within A's URL-space.
2. JC sends an ITML Best Practices request message using C and blocks on the response
3. A handles the request
4. A creates response document
5. A sends response document over C
6. JC returns response to requester
7. JC optionally closes connection C

HTTP Sample:

```
POST /signon HTTPS/1.1
Host: www.jamcracker.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <!—Best practices document →
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Security

Jamcracker is authenticated at a partner web site through the use of HTTP username and password authentication. There is a username/password combination for each Jamcracker/partner interaction. Message security and integrity are handled by HTTPS.

Asynchronous

ITML is a synchronous protocol. There is no mechanism for asynchronous callbacks.

Conformance

Responses to requests must occur within a period that is defined on a per interaction and per message basis. A non-normative value of 2 minutes shall be used.

ISSUE: What conformance tests above and beyond schema validation are required?

ITML Fault Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for ITML Errors.  Edited by David Orchard -->
<schema targetNamespace="http://www.itml.org/ns/messaging" xmlns="http://www.w3.org/1999/XMLSchema"
xmlns:ITML-ERR="http://www.itml.org/ns/messaging">
    <annotation>
        <appinfo>
    XML Schema for ITML fault codes.
    </appinfo>
    </annotation>
    <complexType name="ITMLFaultDetail" derivedBy="extension">
        <!--base="tns:detail"> -->
        <element name="faultcode" type="ITML-ERR:faultcode"/>
        <element name="faultstring" type="string"/>
    </complexType>
    <simpleType name="faultcode" base="string">
        <enumeration value="InvalidXMLDocument"/>
        <enumeration value="ApplicationFailure"/>
        <enumeration value="RequestTimedOut"/>
        <enumeration value="Unauthorized"/>
        <enumeration value="UnknownCommand"/>
        <enumeration value="VersionMismatch"/>
    </simpleType>
</schema>
```

ITML Message Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for ITML Messages information.  Edited by David Orchard -->
<schema targetNamespace="http://www.itml.org/ns/messaging" xmlns="http://www.w3.org/1999/XMLSchema"
xmlns:msg="http://www.itml.org/ns/messaging">
    <annotation>
        <appinfo>
    XML Schema for ITML Messaging.  Particularly transaction IDs.
    </appinfo>
    </annotation>
    <complexType name="txid" base="string">
        <pattern value="[a-Z]{3}:[0-9]{2}:[0-9]{2}:[0-9]{2}:[0-9]{2}"/>
    </complexType>
</schema>
```

Appendix

Issues

1. Version Control
2. Conformance
3. Error Recovery

Futures

- Asynchronous
- Version Control
- Higher Security
- Interface Definition

Important Design notes (non-normative)

Asynchronous operation

Asynchronous callbacks are very useful, but the functionality adds an order of magnitude (potentially greatly increases the complexity) to the implementation and framework. It could significantly raise the bar to adoption by member companies. It is probable that standards bodies such as the W3C XML Protocols Working Group or ebXML will address this issue. It is not the intent of this specification to duplicate core web infrastructure work.

Version Control

Should versioning be dealt with in this specification or should it rely upon an underlying spec, such as SOAP? If done in this specification, how to encode them - different namespace, attribute in requests, attribute in request content, different request/content names.

Typical Use cases:

1) Partner implements Provisioning 1.1 but JC does not
2) Partner implements Provisioning 1.0 and JC implements 1.1

Is this an issue? Shouldn't the partner and JC simply agree when to switch to new versions? It's up to partner and JC to track backwards compatible senders/receivers.

It is probable that standards bodies such as the W3C XML Protocols Working Group or ebXML will address this issue. It is not the intent of this specification to duplicate core web infrastructure work.

Authentication

There are many alternative mechanisms that offer stronger security than username/password. One example is making conversations stateful with a dynamic token passed in each request. It has been determined that the significant extra complexity to provide additional security is not a valued trade-off. Another example is adding extra security information into a header field.

It is probable that standards bodies such as the W3C XML Protocols Working Group or ebXML will address this issue. It is not the intent of this specification to duplicate core web infrastructure work.

Error Handling and Recovery

The specification has avoided any mention of error recovery or increased reliability. Typical examples of these are retrying the transaction, adopting a two-phase commit protocol, defining compensating transactions. Therefore it is likely that this will be a trouble spot for integration.
ISSUE: Is there a minimal set of specifications that can be made to facilitate error handling? For example, if an error is a server error then a retry will happen within 2 hours?

Credits

This specification has been greatly helped by the following people and affiliations:
Viswanath Reddy – Jamcracker
Vinay Singla - Jamcracker
Camie Hackson – ManageMark
Chris Haddad – EmployEase
Jon Finegold – OpenAir

Revision History

2000:

July 20 – Initial Revision.
July 26 – Added ebXML reference. Added support for ebXML when SOAP-style documents with separate DTDs cannot be used.
Aug 3 to 15 – Many revisions
Sep 30 to Oct 6 – Minor revisions
Oct 10 – Added Partner specific namespaces to support partner defined roles.
Nov 1 – Added Transaction ID
Nov 20 – Added Encoding rules